FIG. 1 PRIOR ART

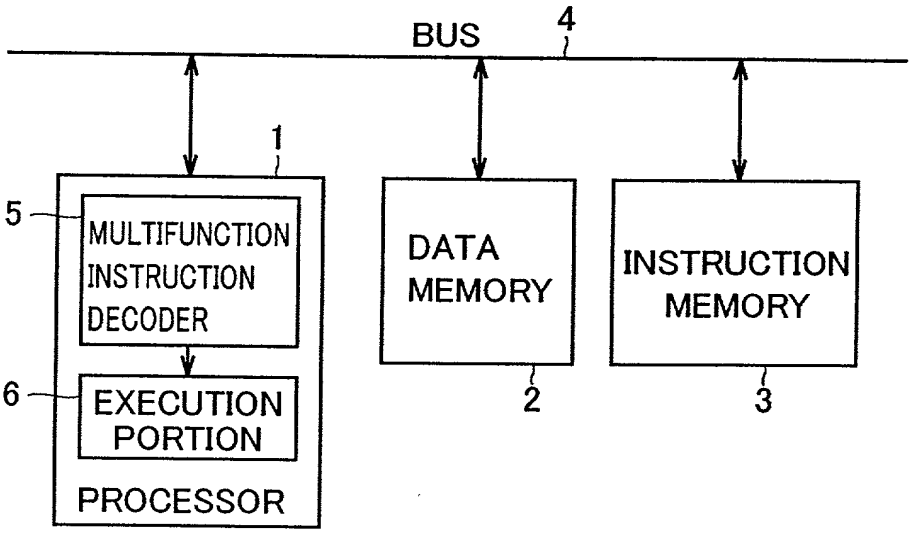


FIG. 2

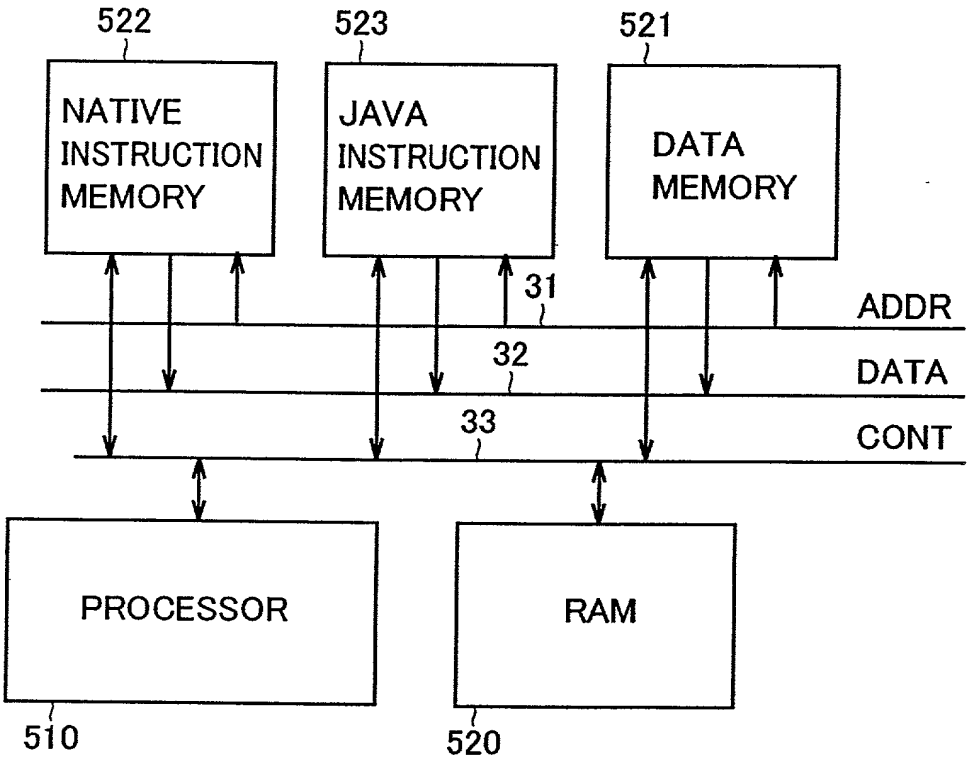
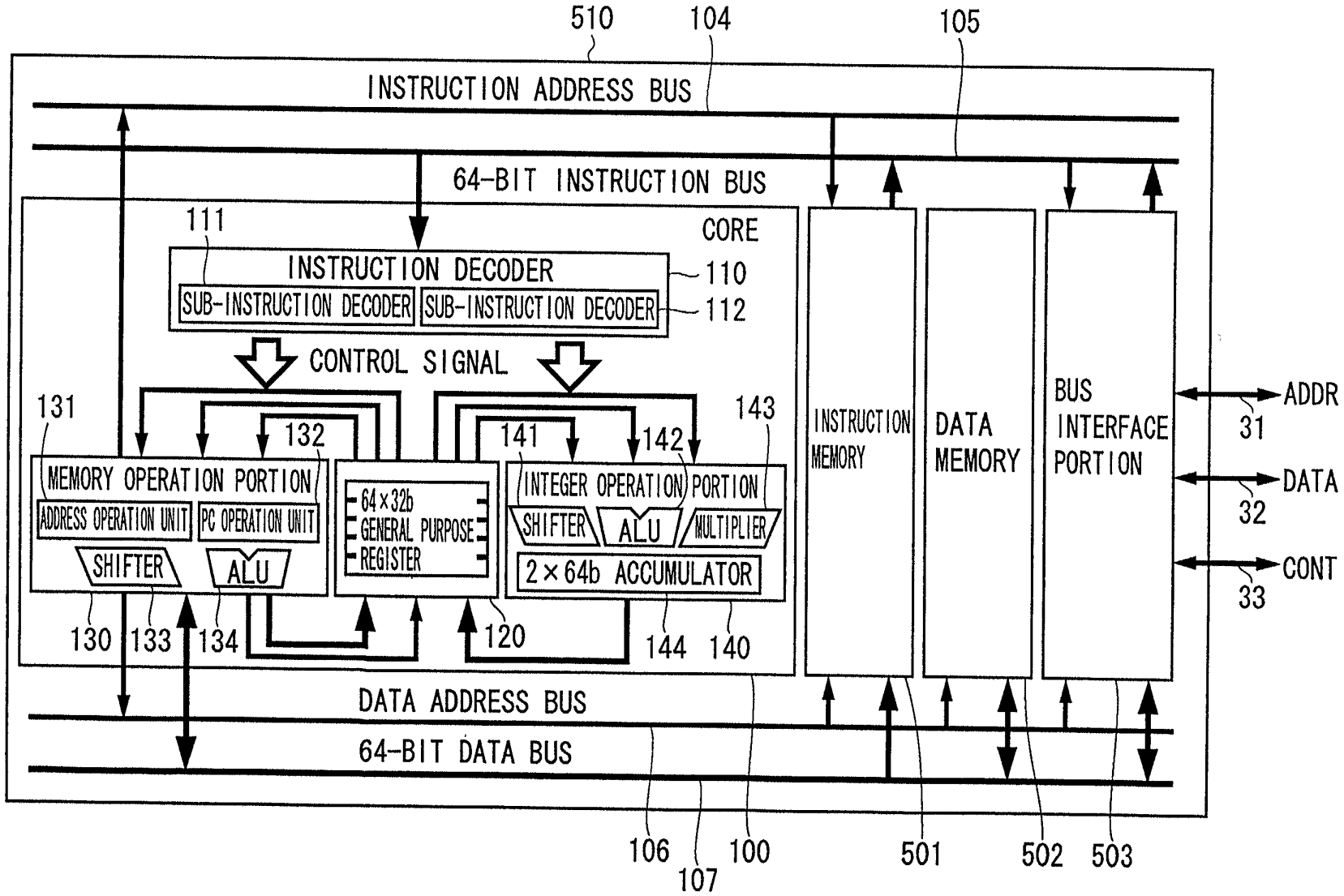


FIG. 3


510
104
105
INSTRUCTION ADDRESS BUS
64-BIT INSTRUCTION BUS
111
CORE
INSTRUCTION DECODER
110
SUB-INSTRUCTION DECODER
SUB-INSTRUCTION DECODER
112
CONTROL SIGNAL
131
132
141
142
143
MEMORY OPERATION PORTION
ADDRESS OPERATION UNIT
PC OPERATION UNIT
SHIFTER
ALU
64×32b GENERAL PURPOSE REGISTER
INTEGER OPERATION PORTION
SHIFTER
ALU
MULTIPLIER
2×64b ACCUMULATOR
INSTRUCTION MEMORY
DATA MEMORY
BUS INTERFACE PORTION
ADDR
31
DATA
32
CONT
33
130
133
134
120
144
140
DATA ADDRESS BUS
64-BIT DATA BUS
106
107
100
501
502
503

US5220,682,660

FIG. 4

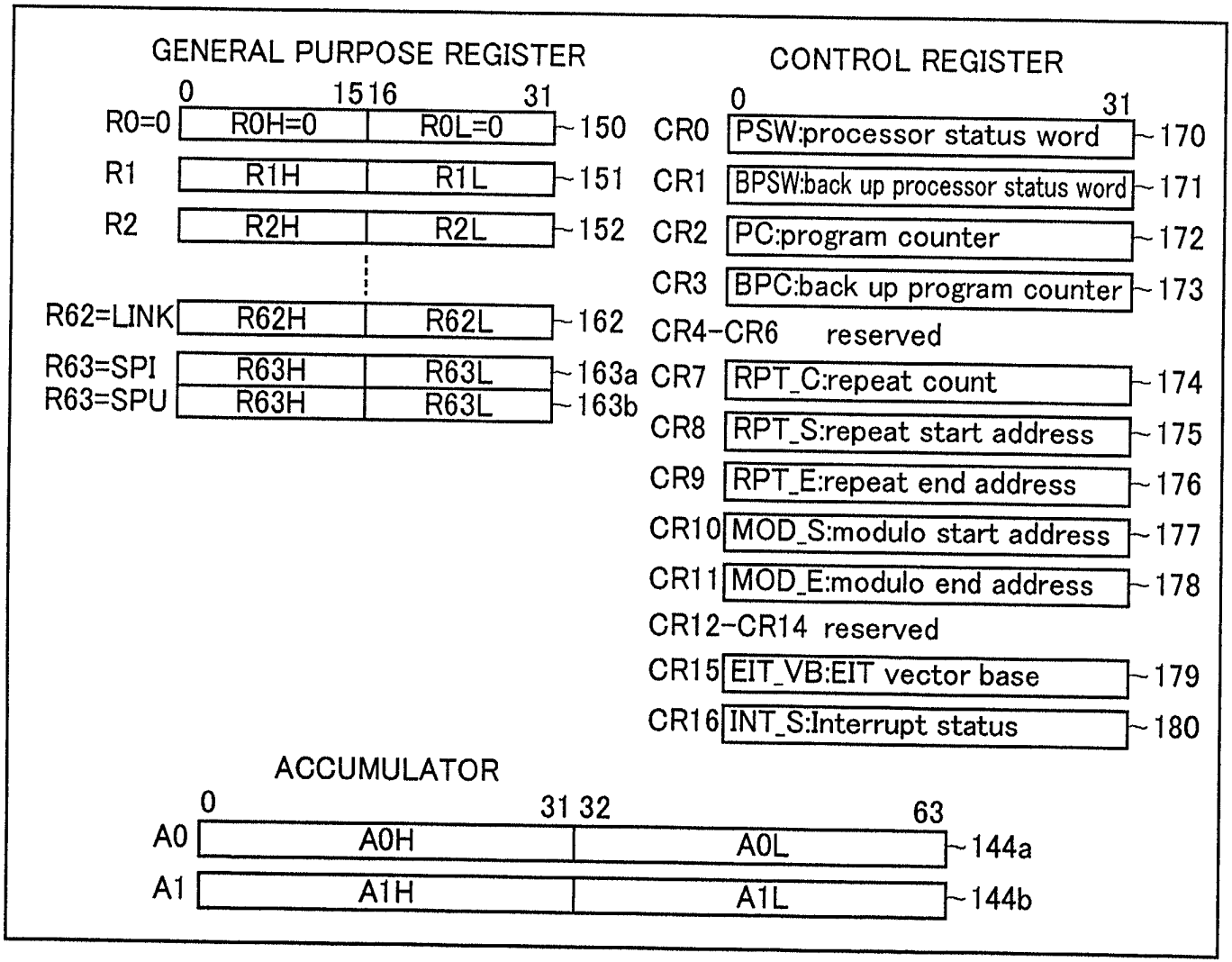
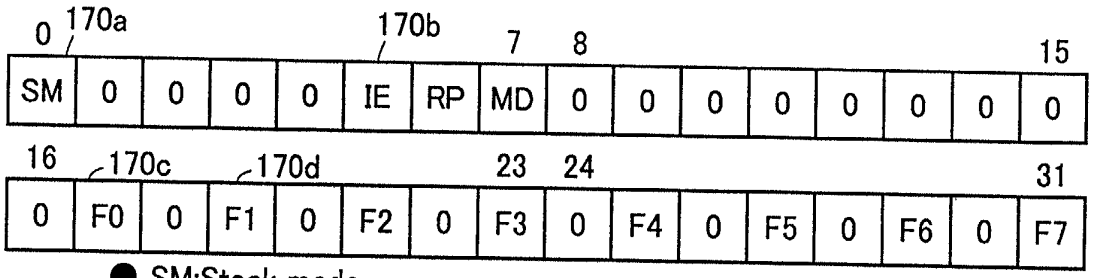


FIG. 5



- SM:Stack mode
  SM=0 Interrupt mode.SPI is used.
  SM=1 User mode.SPU is used.
- IE:Interrupt enable
  IE=0 Interrupts are masked.
  IE=1 Interrupts are accepted.
- RP:Repeat enable
  RP=0 A block repeat is inactive.
  RP=1 A block repeat is active.
- MD:Modulo enable
  MD=0 Modulo addressing is disabled.
  MD=1 Modulo addressing is enabled.
- F0 :Execution control flag #0
- F1 :Execution control flag #1
- F2 :General flag
- F3 :General flag
- F4(S) :Saturation flag
- F5(V) :Overflow flag
- F6(VA) :Accumulated Overflow flag
  VA is cleared by a reset interrupt and MVTSYS instruction.
- F7(C) :Carry/Borrow flag

FIG. 6

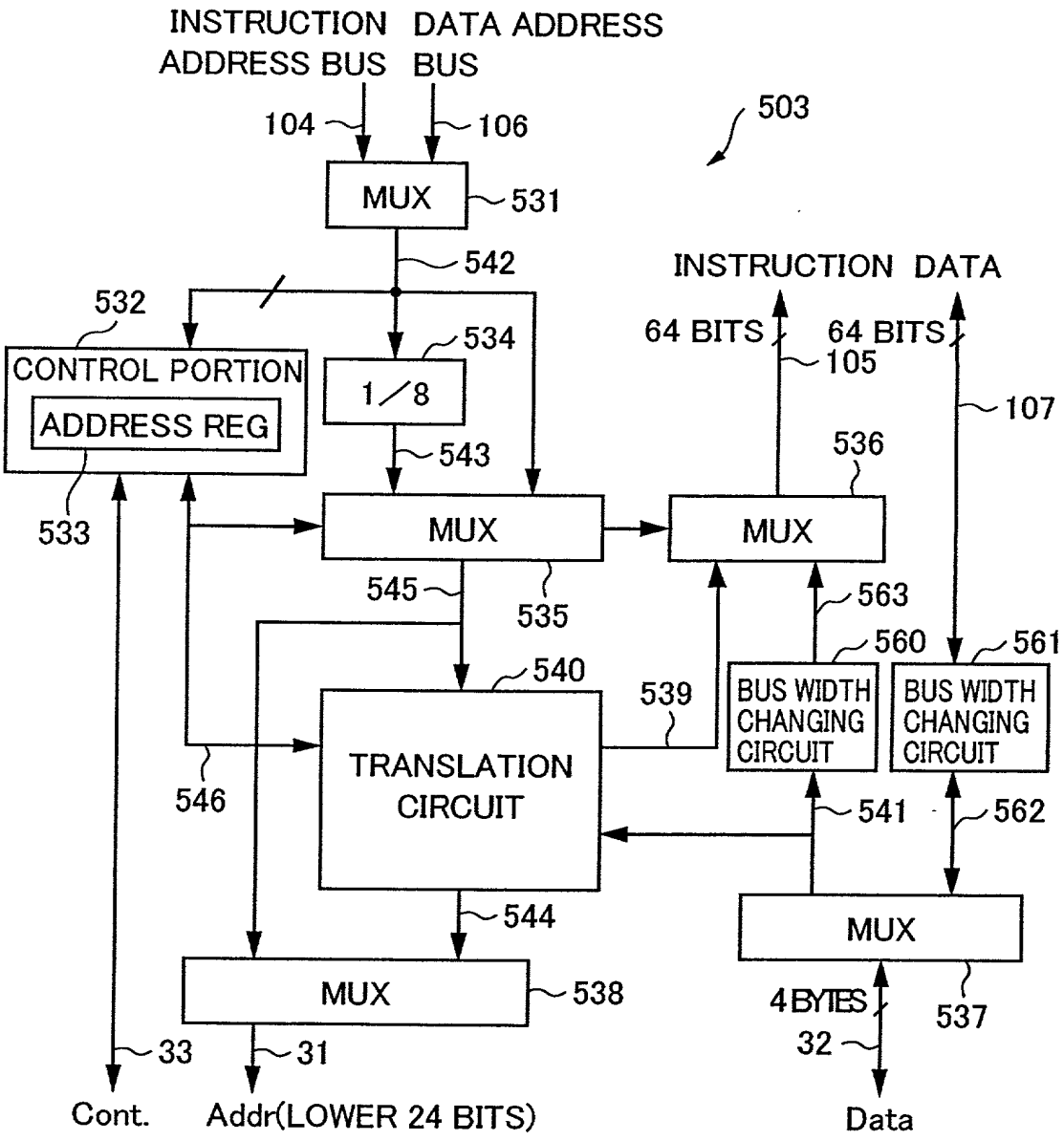

INSTRUCTION ADDRESS BUS
DATA ADDRESS BUS
104
106
503
MUX
531
542
INSTRUCTION DATA
64 BITS
64 BITS
105
107
532
CONTROL PORTION
ADDRESS REG
534
1/8
543
536
533
MUX
MUX
545
535
563
560
561
540
539
BUS WIDTH CHANGING CIRCUIT
BUS WIDTH CHANGING CIRCUIT
546
TRANSLATION CIRCUIT
541
562
544
MUX
MUX
538
4BYTES
537
32
33
31
Cont.
Addr(LOWER 24 BITS)
Data

FIG. 7

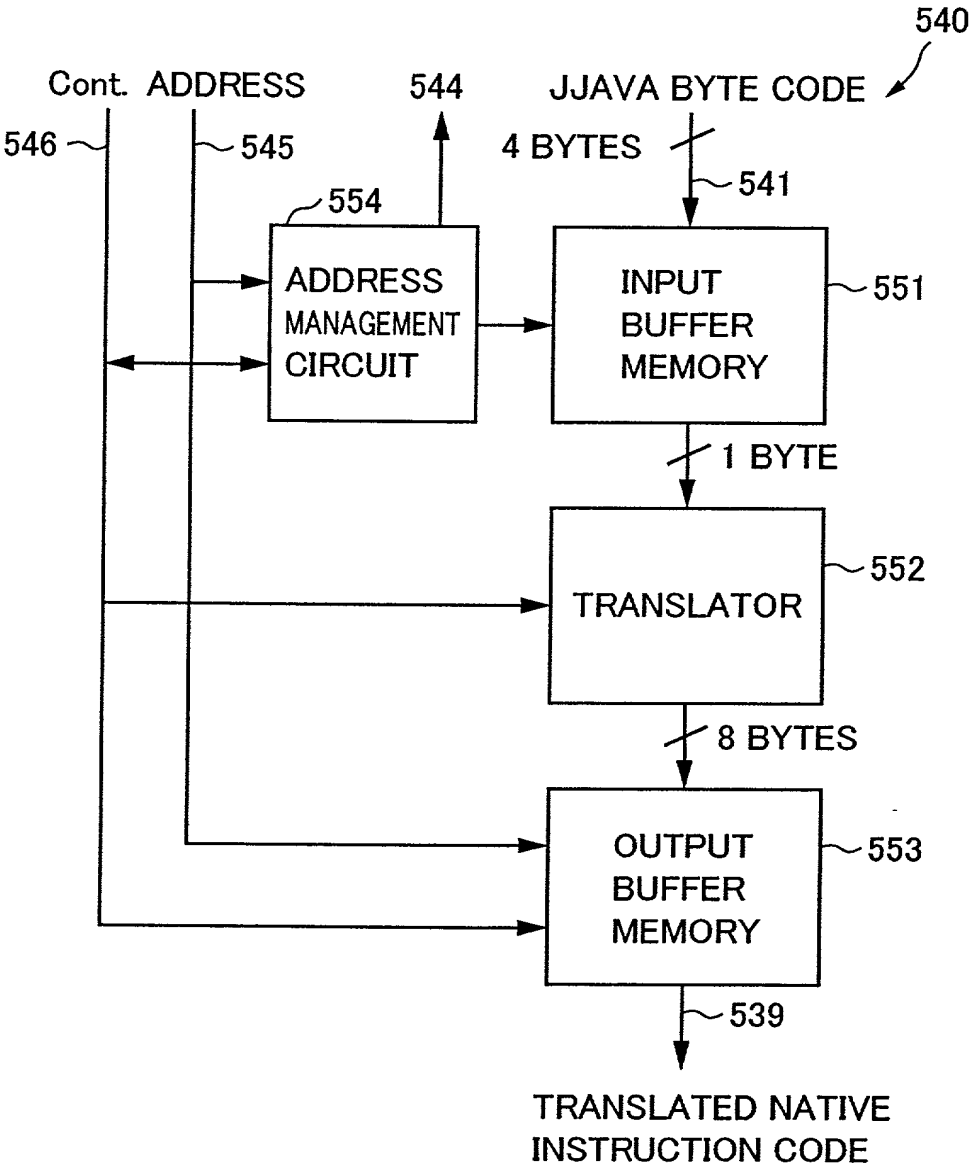
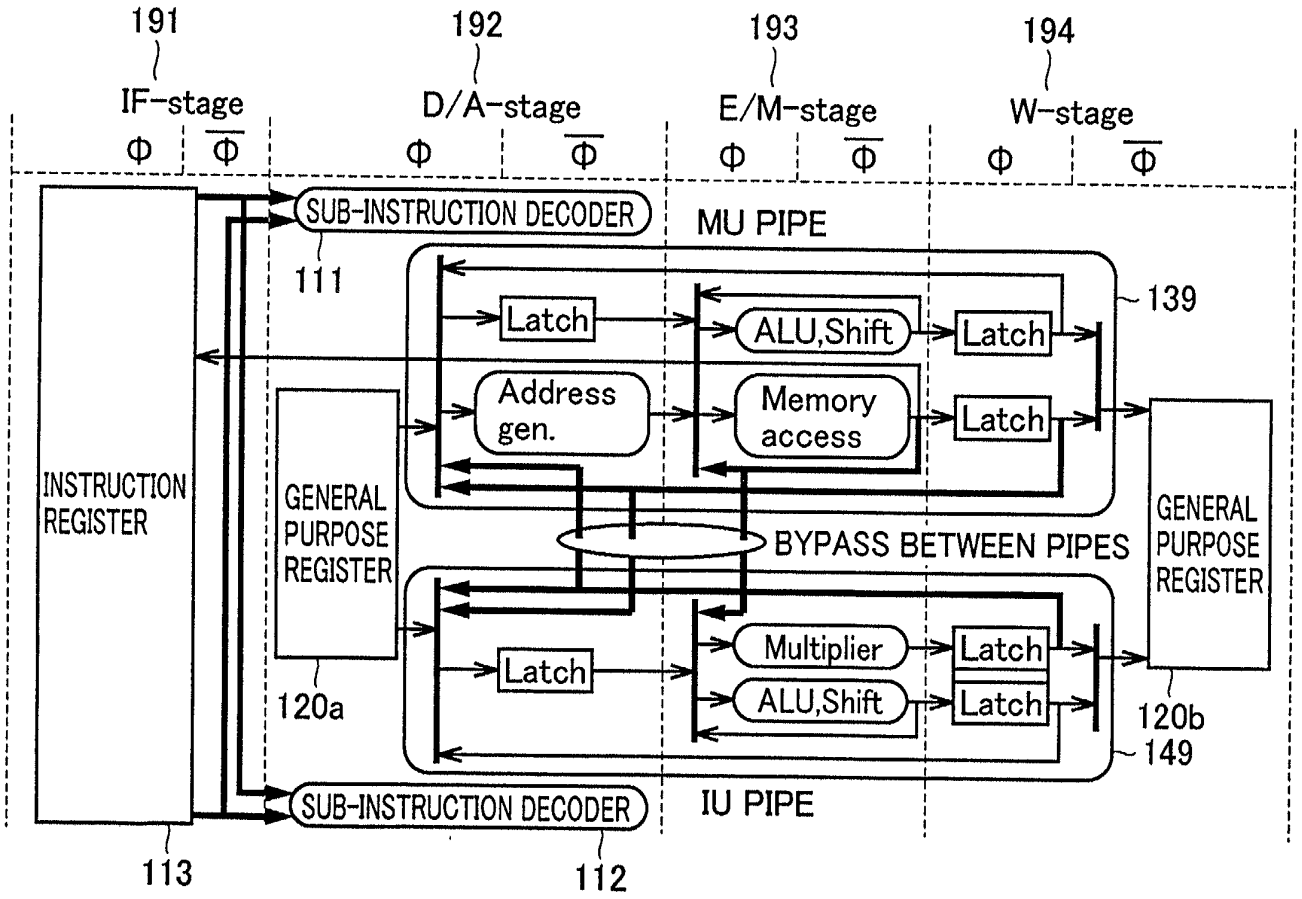
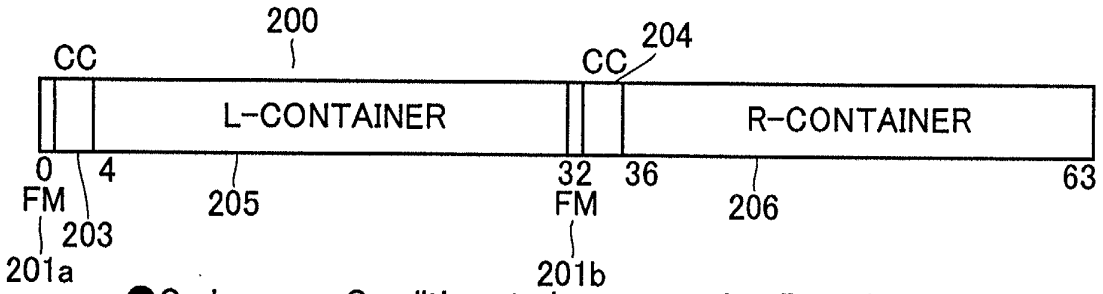

540
Cont. ADDRESS
544
JJAVA BYTE CODE
546
545
4 BYTES
541
554
ADDRESS MANAGEMENT CIRCUIT
INPUT BUFFER MEMORY
551
1 BYTE
TRANSLATOR
552
8 BYTES
OUTPUT BUFFER MEMORY
553
539
TRANSLATED NATIVE INSTRUCTION CODE

FIG. 8


191
192
193
194
IF-stage
D/A-stage
E/M-stage
W-stage
Φ
$\overline{\Phi}$
SUB-INSTRUCTION DECODER
MU PIPE
111
139
Latch
ALU,Shift
Address gen.
Memory access
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTER
BYPASS BETWEEN PIPES
Multiplier
120a
120b
149
IU PIPE
113
112

T0520 "6E7TT660

FIG. 9



Code	Conditions to be executed	Description
CC = 000	Always	---
CC = 001	F0 = T and F1 = don't care	/TX
CC = 010	F0 = F and F1 = don't care	/FX
CC = 011	F0 = don't care and F1 = T	/XT
CC = 100	F0 = don't care and F1 = F	/XF
CC = 101	F0 = T and F1 = T	/TT
CC = 110	F0 = T and F1 = F	/TF
CC = 111	Reserved	---

FM	NUMBER OF OUTPUT SUB-INSTRUCTIONS	OUTPUT ORDER L-CONTAINER	OUTPUT ORDER R-CONTAINER
00	two	1st	1st
01	two	1st	2nd
10	two	2nd	1st
11	one	1st	----

FIG. 10

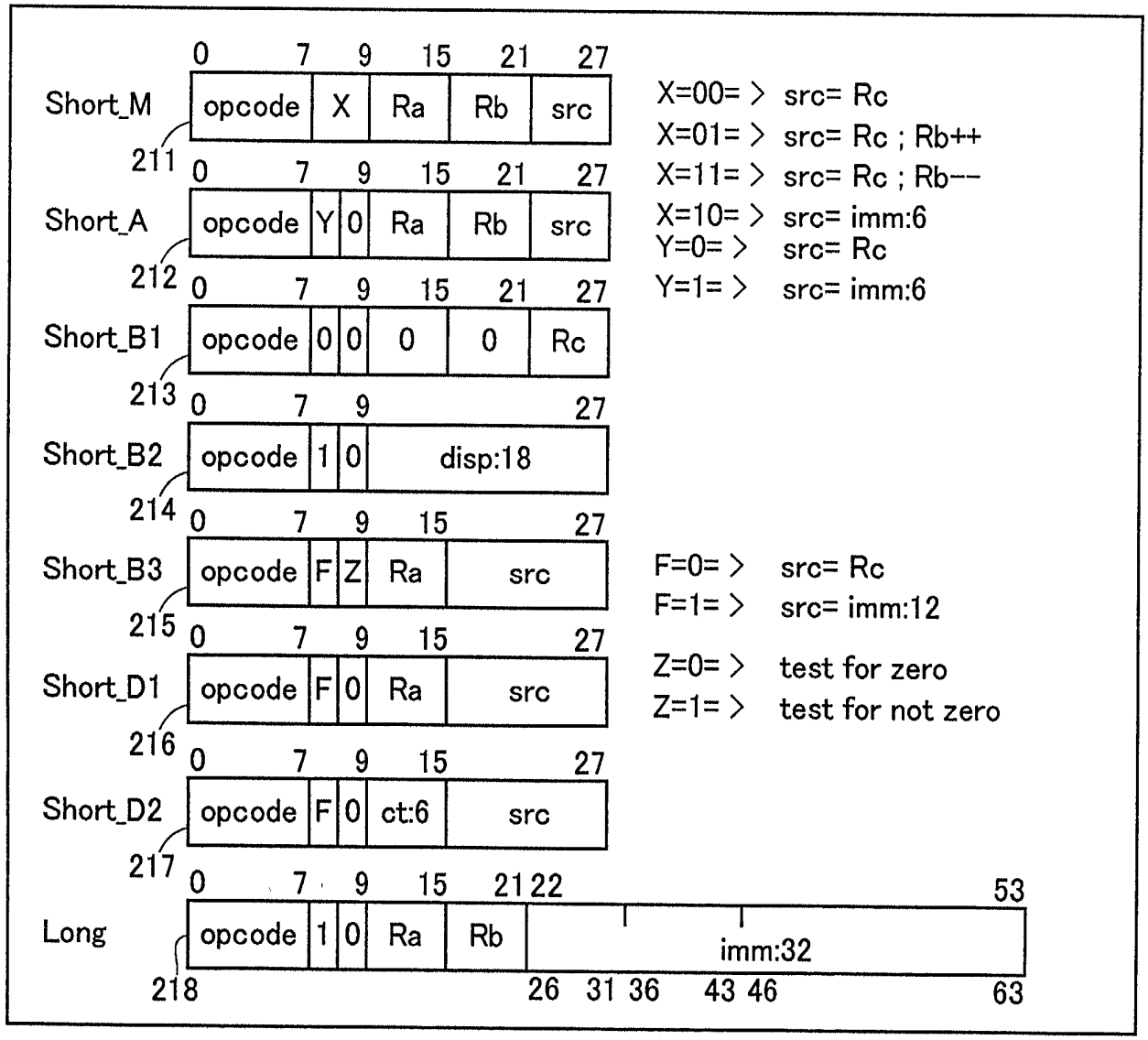

0 7 9 15 21 27
Short_M
opcode X Ra Rb src
211
0 7 9 15 21 27
Short_A
opcode Y 0 Ra Rb src
212
0 7 9 15 21 27
Short_B1
opcode 0 0 0 0 Rc
213
0 7 9 27
Short_B2
opcode 1 0 disp:18
214
0 7 9 15 27
Short_B3
opcode F Z Ra src
215
0 7 9 15 27
Short_D1
opcode F 0 Ra src
216
0 7 9 15 27
Short_D2
opcode F 0 ct:6 src
217
0 7 9 15 21 22 53
Long
opcode 1 0 Ra Rb imm:32
218
26 31 36 43 46 63
X=00= > src= Rc
X=01= > src= Rc ; Rb++
X=11= > src= Rc ; Rb--
X=10= > src= imm:6
Y=0= > src= Rc
Y=1= > src= imm:6
F=0= > src= Rc
F=1= > src= imm:12
Z=0= > test for zero
Z=1= > test for not zero

FIG. 11

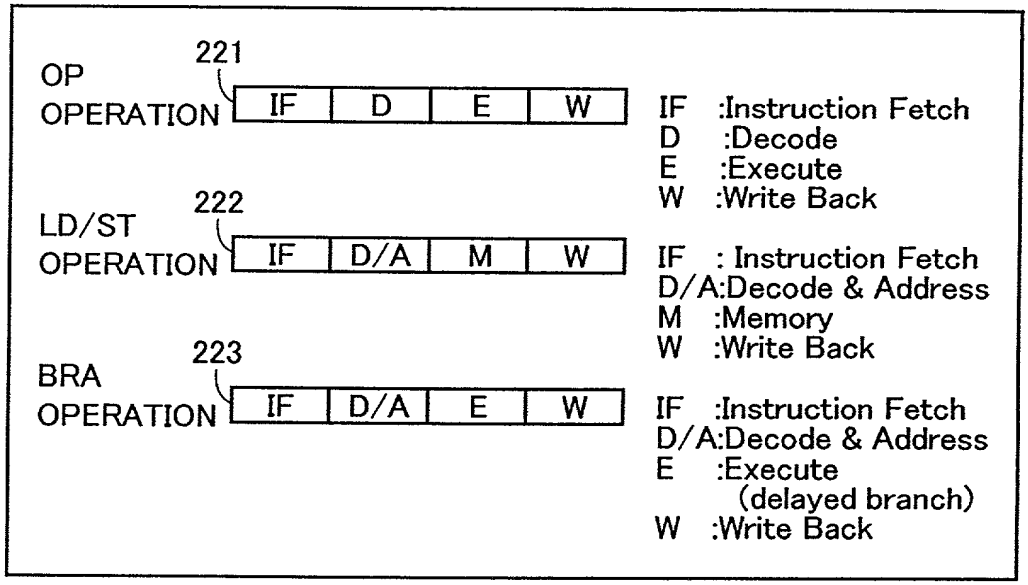

221
OP
OPERATION
IF
D
E
W
IF :Instruction Fetch
D :Decode
E :Execute
W :Write Back
222
LD/ST
OPERATION
IF
D/A
M
W
IF : Instruction Fetch
D/A:Decode & Address
M :Memory
W :Write Back
223
BRA
OPERATION
IF
D/A
E
W
IF :Instruction Fetch
D/A:Decode & Address
E :Execute
(delayed branch)
W :Write Back

FIG. 12

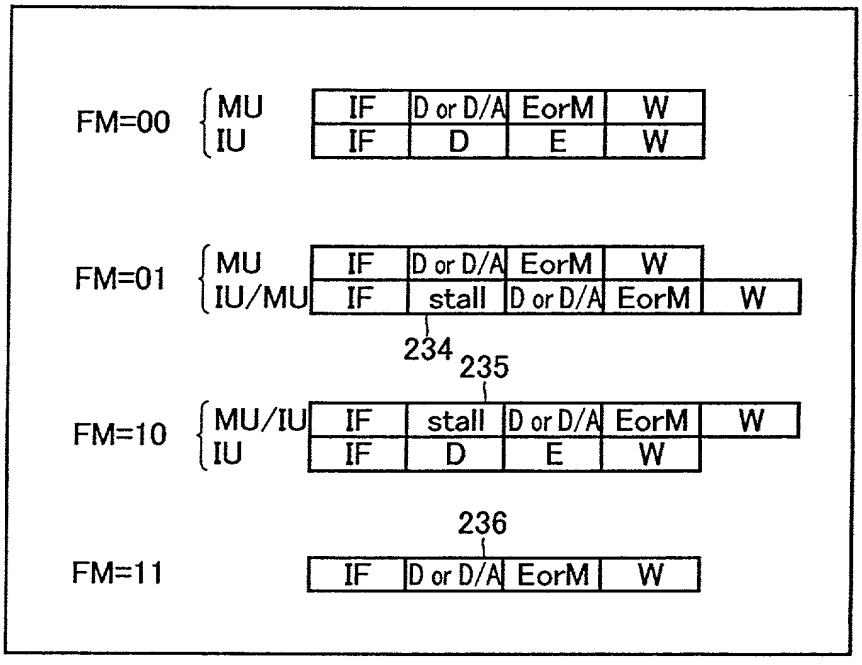

FM=00
MU
IU
IF
D or D/A
EorM
W
IF
D
E
W
FM=01
MU
IU/MU
IF
D or D/A
EorM
W
IF
stall
D or D/A
EorM
W
234
235
FM=10
MU/IU
IU
IF
stall
D or D/A
EorM
W
IF
D
E
W
236
FM=11
IF
D or D/A
EorM
W

FIG. 13

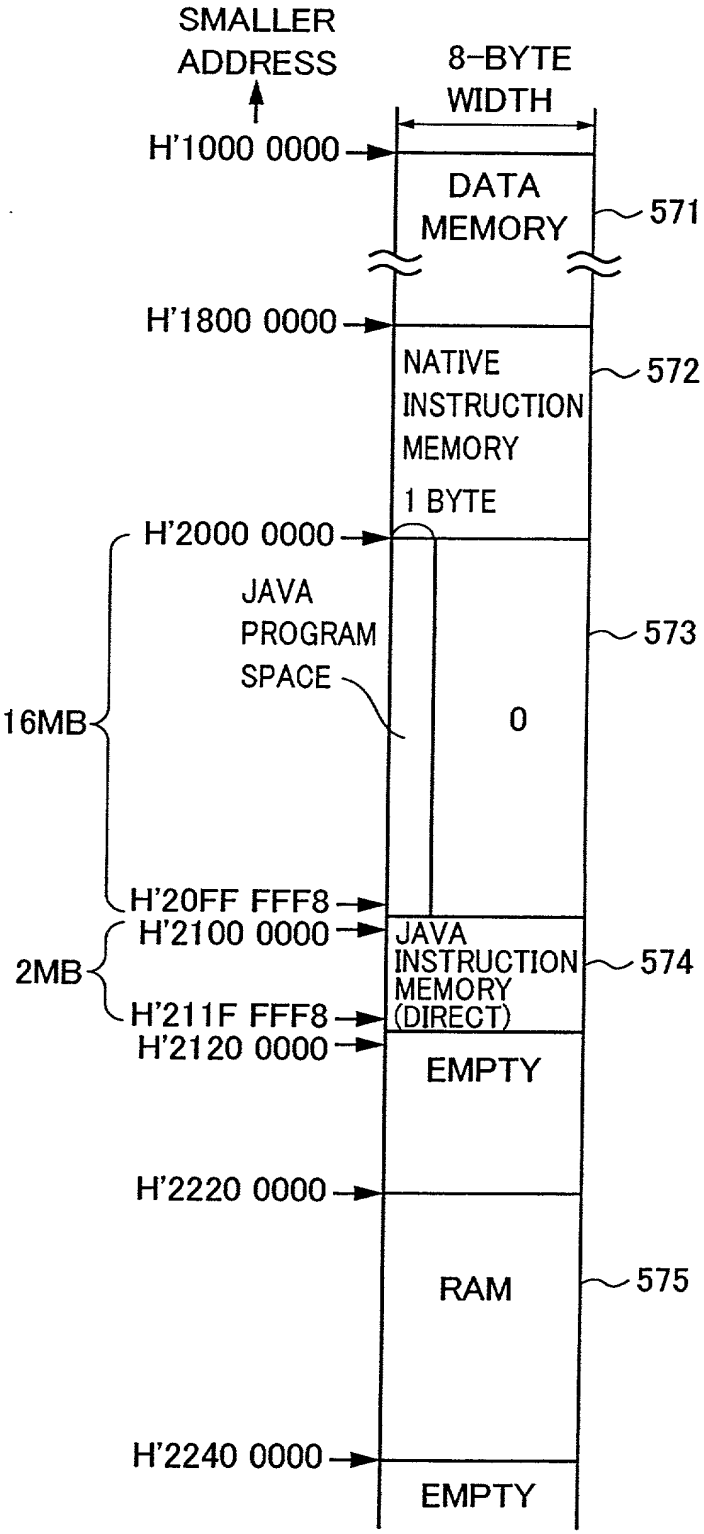

SMALLER ADDRESS
8-BYTE WIDTH
H'1000 0000
DATA MEMORY
571
H'1800 0000
NATIVE INSTRUCTION MEMORY
572
1 BYTE
H'2000 0000
JAVA PROGRAM SPACE
573
16MB
0
H'20FF FFF8
H'2100 0000
JAVA INSTRUCTION MEMORY (DIRECT)
574
2MB
H'211F FFF8
H'2120 0000
EMPTY
H'2220 0000
RAM
575
H'2240 0000
EMPTY

FIG. 14

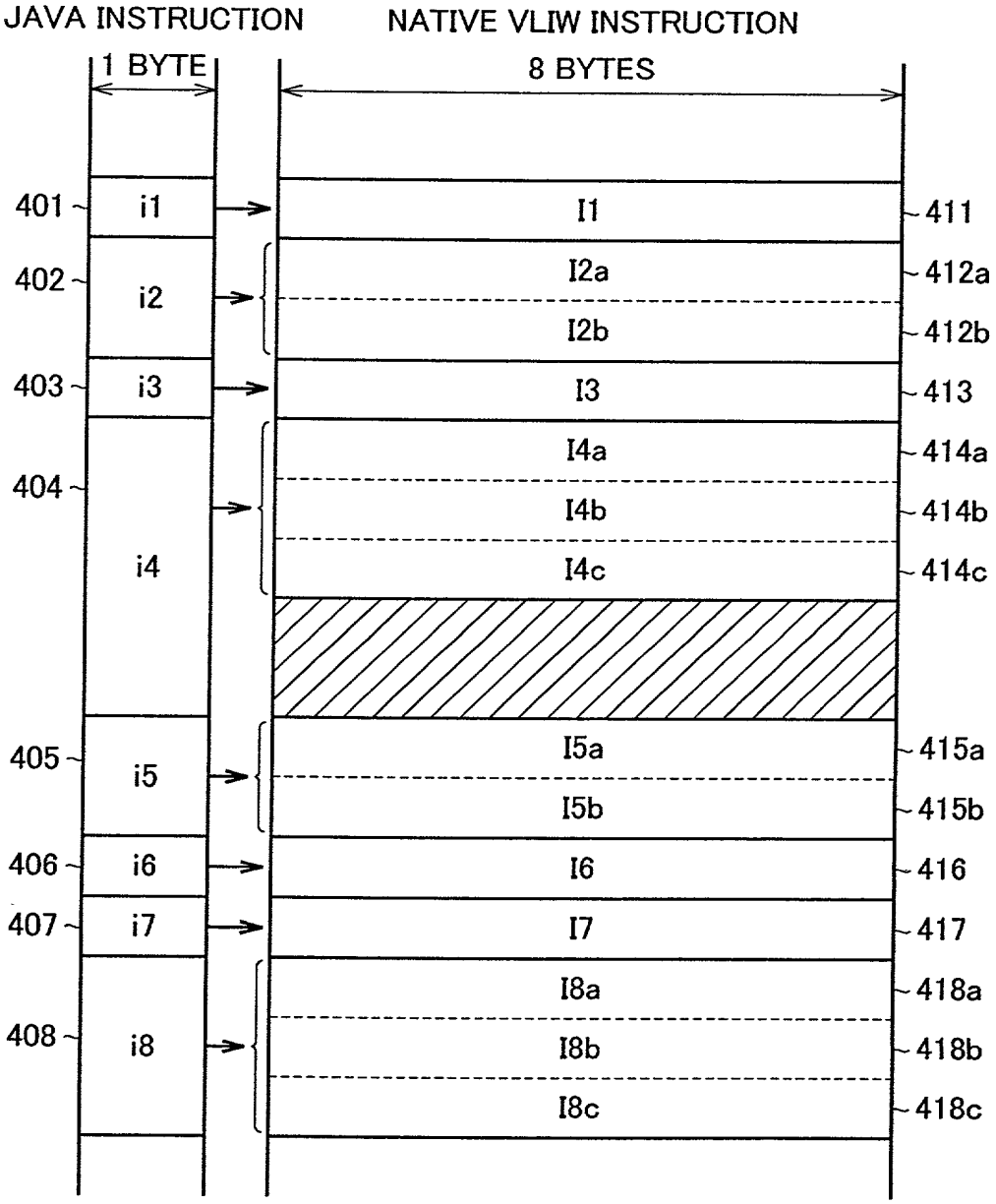

JAVA INSTRUCTION
NATIVE VLIW INSTRUCTION
1 BYTE
8 BYTES
401
i1
I1
411
402
i2
I2a
412a
I2b
412b
403
i3
I3
413
404
i4
I4a
414a
I4b
414b
I4c
414c
405
i5
I5a
415a
I5b
415b
406
i6
I6
416
407
i7
I7
417
408
i8
I8a
418a
I8b
418b
I8c
418c

FIG. 15

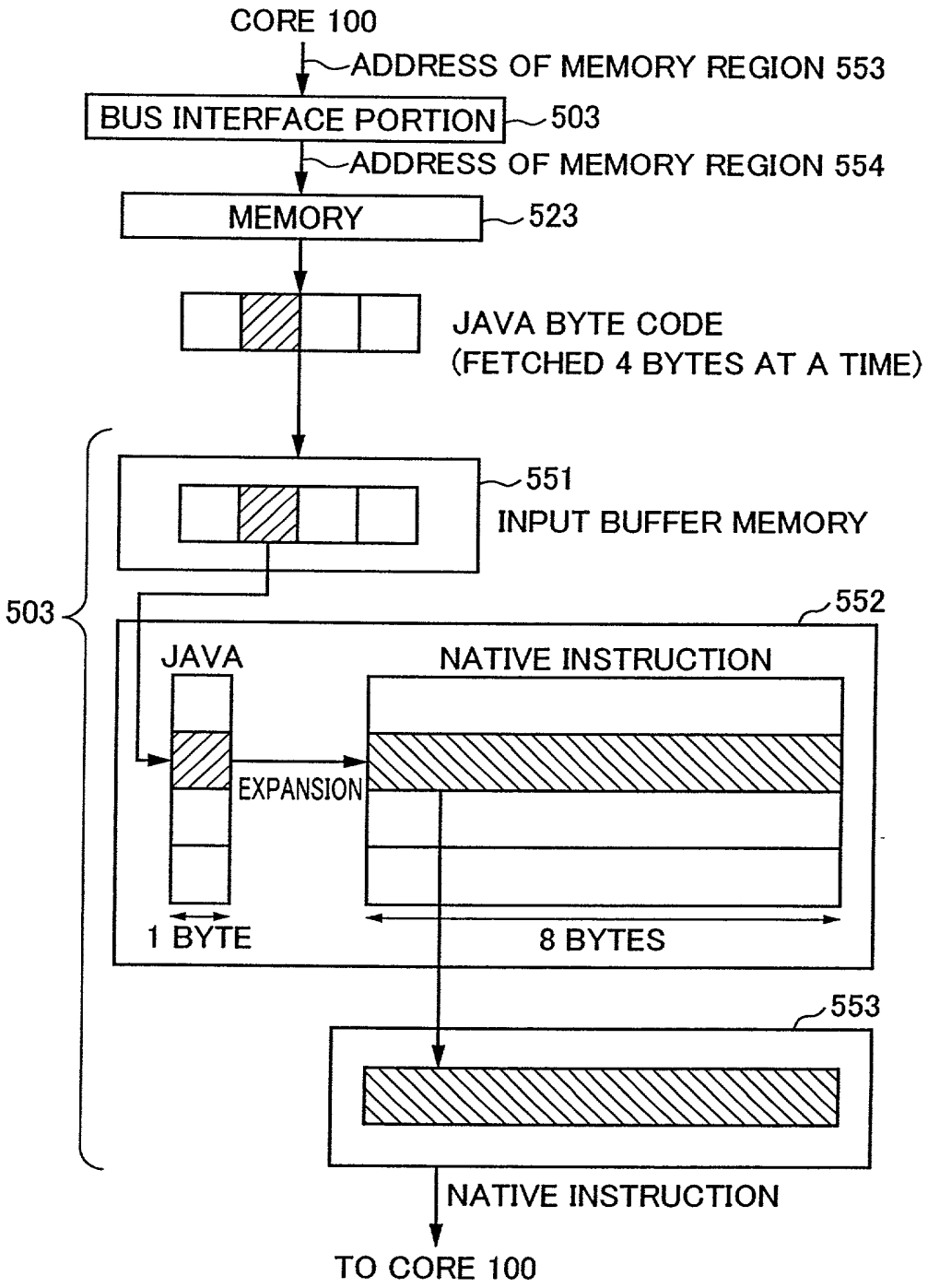
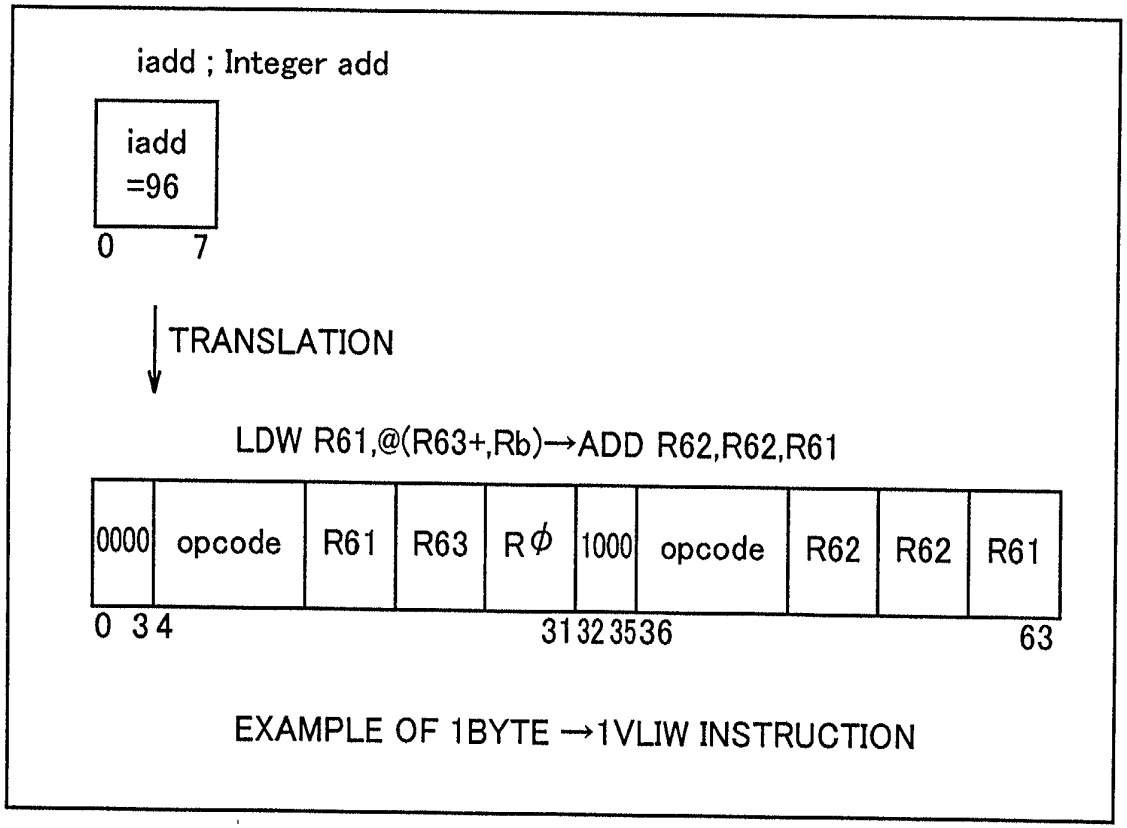

CORE 100
ADDRESS OF MEMORY REGION 553
BUS INTERFACE PORTION
503
ADDRESS OF MEMORY REGION 554
MEMORY
523
JAVA BYTE CODE
(FETCHED 4 BYTES AT A TIME)
551
INPUT BUFFER MEMORY
503
552
JAVA
NATIVE INSTRUCTION
EXPANSION
1 BYTE
8 BYTES
553
NATIVE INSTRUCTION
TO CORE 100

FIG. 16


iadd ; Integer add
iadd
=96
0 7
TRANSLATION
LDW R61,@(R63+,Rb)→ADD R62,R62,R61
0000
opcode
R61
R63
Rϕ
1000
opcode
R62
R62
R61
0 3 4
31 32 35 36
63
EXAMPLE OF 1BYTE →1VLIW INSTRUCTION

T05220"6E7T660

FIG. 17

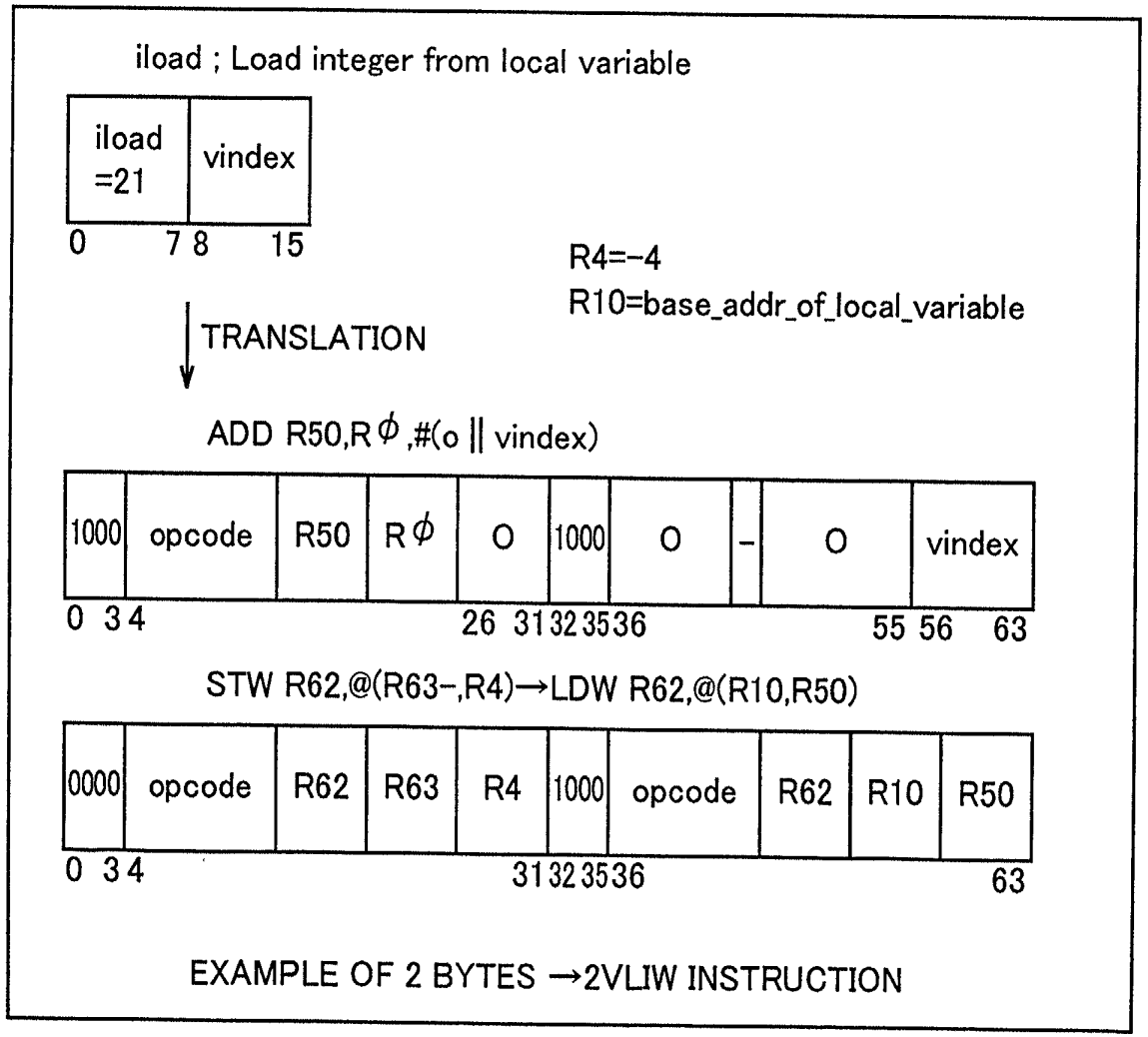


FIG. 18

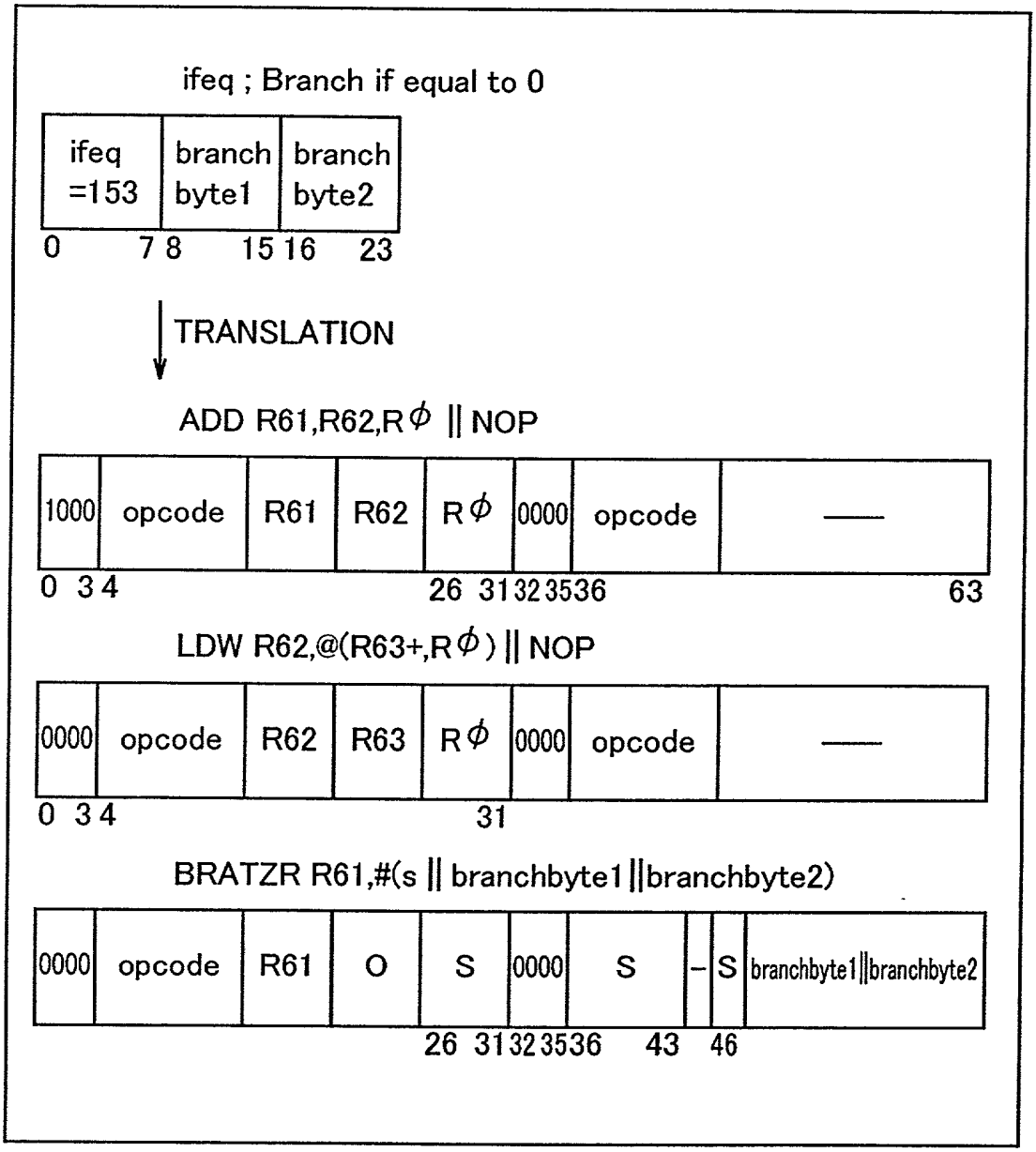

ifeq ; Branch if equal to 0
ifeq =153
branch byte1
branch byte2
0 7 8 15 16 23
TRANSLATION
ADD R61,R62,Rϕ || NOP
1000
opcode
R61
R62
Rϕ
0000
opcode
——
0 3 4 26 31 32 35 36 63
LDW R62,@(R63+,Rϕ) || NOP
0000
opcode
R62
R63
Rϕ
0000
opcode
——
0 3 4 31
BRATZR R61,#(s || branchbyte1 ||branchbyte2)
0000
opcode
R61
O
S
0000
S
–
S
branchbyte1||branchbyte2
26 31 32 35 36 43 46

FIG. 19

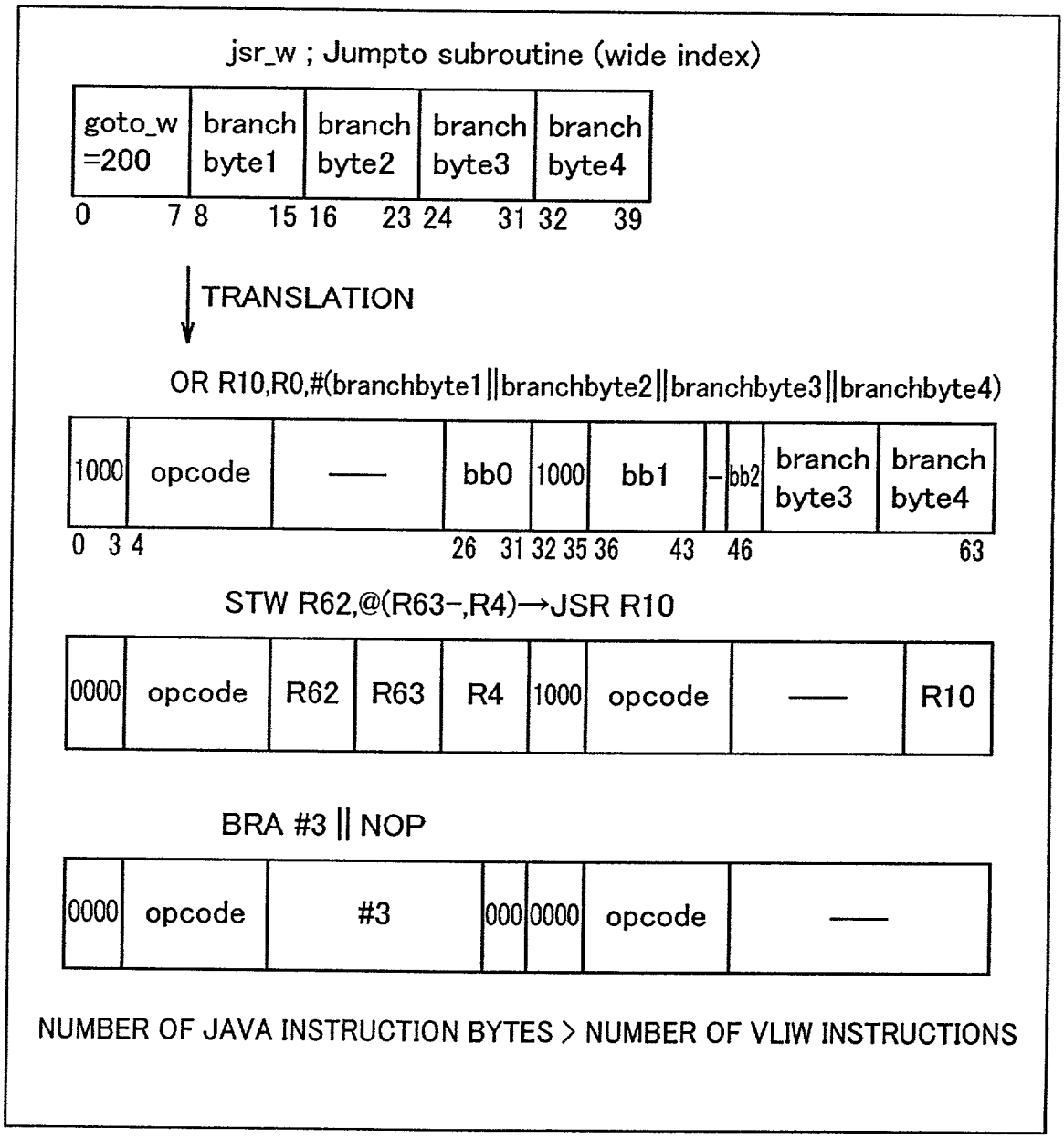

jsr_w ; Jumpto subroutine (wide index)
goto_w =200
branch byte1
branch byte2
branch byte3
branch byte4
0 7 8 15 16 23 24 31 32 39
TRANSLATION
OR R10,R0,#(branchbyte1||branchbyte2||branchbyte3||branchbyte4)
1000
opcode
——
bb0
1000
bb1
–
bb2
branch byte3
branch byte4
0 3 4 26 31 32 35 36 43 46 63
STW R62,@(R63–,R4)→JSR R10
0000
opcode
R62
R63
R4
1000
opcode
——
R10
BRA #3 || NOP
0000
opcode
#3
000
0000
opcode
——
NUMBER OF JAVA INSTRUCTION BYTES > NUMBER OF VLIW INSTRUCTIONS

FIG. 20

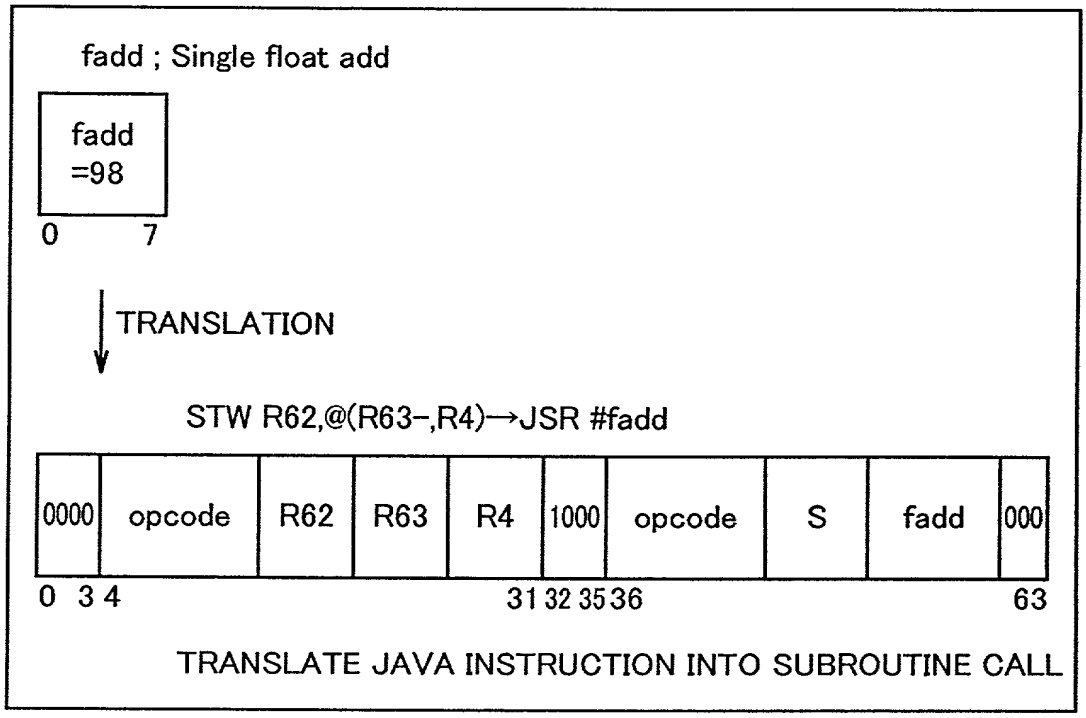
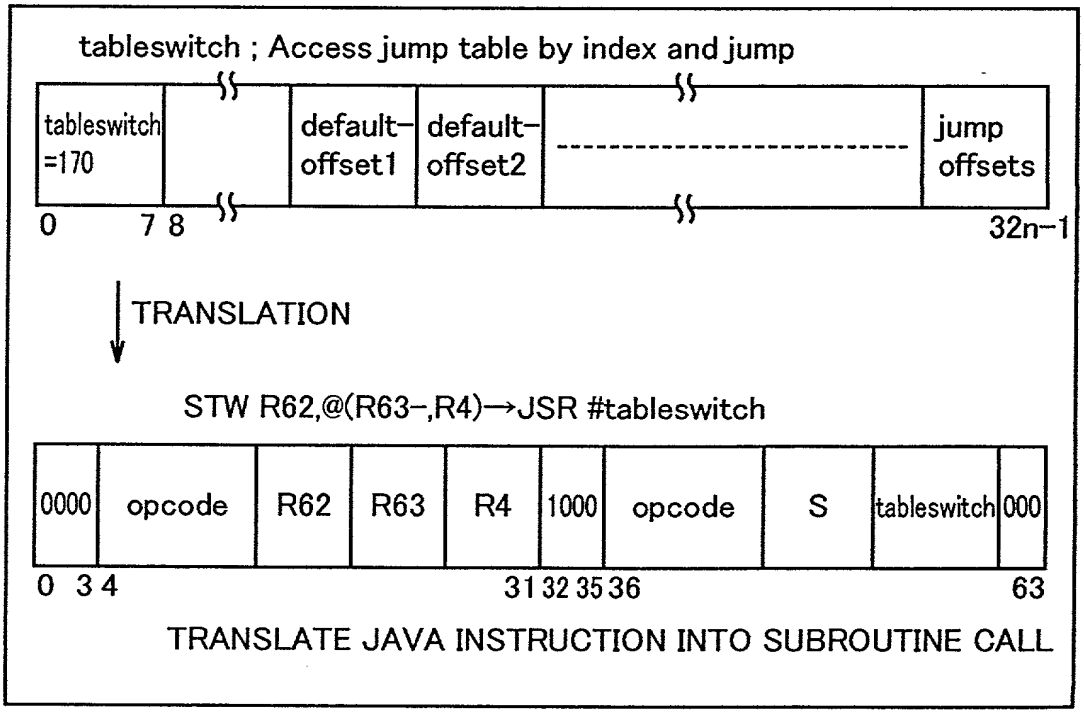

fadd ; Single float add
fadd
=98
0 7
TRANSLATION
STW R62,@(R63-,R4)→JSR #fadd
0000
opcode
R62
R63
R4
1000
opcode
S
fadd
000
0 3 4
31 32 35 36
63
TRANSLATE JAVA INSTRUCTION INTO SUBROUTINE CALL

FIG. 21


tableswitch ; Access jump table by index and jump
tableswitch
=170
default-
offset1
default-
offset2
jump
offsets
0 7 8
32n-1
TRANSLATION
STW R62,@(R63-,R4)→JSR #tableswitch
0000
opcode
R62
R63
R4
1000
opcode
S
tableswitch
000
0 3 4
31 32 35 36
63
TRANSLATE JAVA INSTRUCTION INTO SUBROUTINE CALL